

Chapter 11

사전학습 모델

사전학습 모델

사전학습 모델: 특정 목적을 가지고, 대규모 데이터로 미리 학습시켜 놓은 모델을 사전학습 모델 이라고 한다.

✓ 이번 실습에서는 다양한 클래스의 이미지로 구성된 ImageNet 데이터로 사전학습 된 합성곱 기반 모델들을 가지고 사전학습 모델에 대해 실습할 것이다.

데이터 전처리 위한 모드 설정



ImageNet

사전학습하기 위한 훈련
데이터

- 2만 개 이상의 클래스
- 총 14,197,122 장의 이미지로 구성된 이미지 데이터셋

VGG16, VGG19, ResNet

mode='caffe'

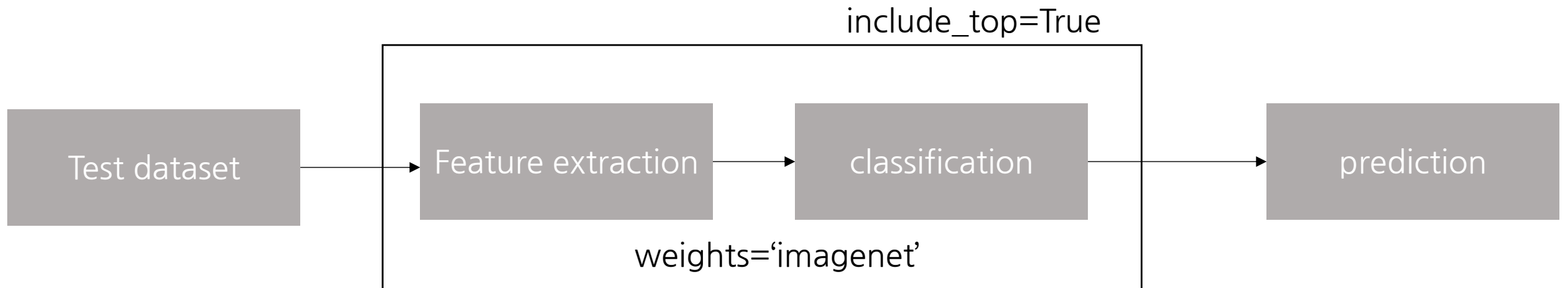
ResNet50V2, InceptionV3,
InceptionResNetV2, Xception

mode='tf'

사전학습 모델

- 사전학습 모델에서 include_top=True 이면, 분류를 위한 완전 연결층을 포함한 모델을 생성하고, weights='imagenet' 을 주면 imagenet 데이터로 사전학습 해서 얻어낸 가중치를 받아와서 사용할 수 있다.
- 이미 학습된 가중치들이기 때문에, 별 다른 추가 학습 없이 imagenet의 1,000 개 테스트 이미지 데이터를 분류할 수 있다.

〈사전학습된 합성곱 신경망〉

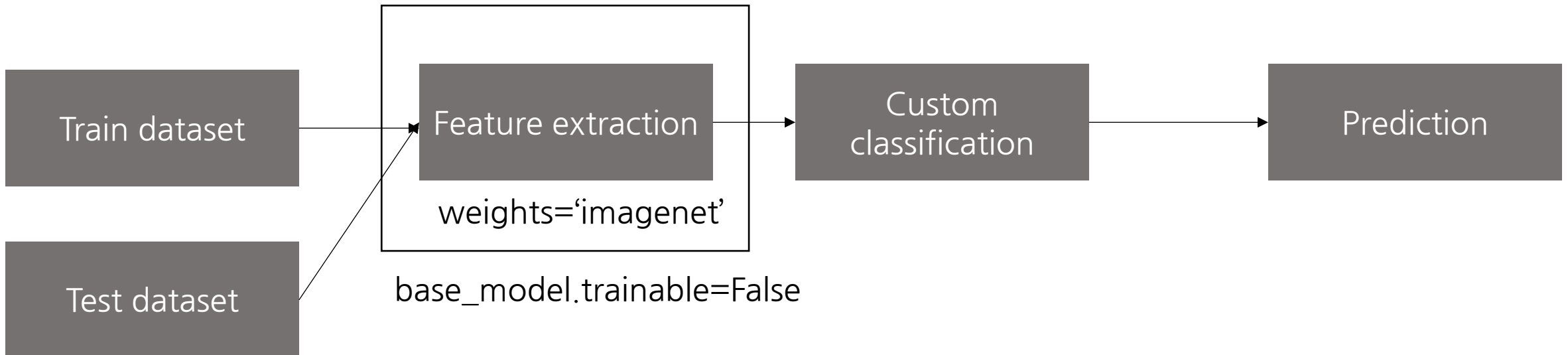


사전학습 모델

- 만약 `include_top=False` 이면, 특징추출 부분은 포함하고(합성곱 층 만 포함), 분류를 위한 완전 연결층은 포함하지 않은 기본모델을 생성한다.
- 한편 기본 모델은 사전학습 된 imagenet 가중치를 사용하고, 완전 연결층 여러 개를 기본 모델에 붙인 다음, `base_model.trainable=False` 로 줘서 기본 모델의 imagenet 가중치를 동결할 수 있다.
- 이후 사용자의 훈련 데이터로 완전 연결층의 가중치만 학습해서 모델 생성하는 방법을 '전이학습(Transfer Learning)' 이라고 한다.

〈사전학습된 합성곱 신경망: 전이학습〉

`include_top=False`



사전학습 모델1: VGG 모델

✓ VGG 모델 시리즈는 합성곱 신경망(CNN) 모델의 일종이다.

VGG16 모델 구성:

- 총 16개 가중치 층(합성곱 층, 완전 연결층)으로 구성
- 합성곱 층 총 13 개 (커널 사이즈=(3,3), 슬라이드 =(1,1))
- 완전 연결층 총 3개
- 풀링, 평탄화 층까지 포함한 모델 총 깊이는 23 층이다.

VGG19 모델 구성:

- VGG16 에 3개의 합성곱 층이 추가된 형태다.
- 따라서 모델 총 깊이는 26 층이다.
- VGG 모델은 224*224 사이즈, RGB 3개 채널로 구성된 이미지를 입력 데이터로 받는다.
- VGG 모델의 전처리 함수 preprocess_input() 은 mode = 'caffe' 가 기본 세팅이다. mode = 'caffe' 로 세팅하면 이미지의 채널 순서를 RGB 순에서 BGR 순으로 바꾸어 준다.

사전학습 모델1: VGG 모델

VGG모델: 데이터 전처리

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import cifar10
##from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.applications.imagenet_utils import preprocess_input

#1: RGB
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
x_train = x_train.astype('float32') # (50000, 32, 32, 3)

#2: default mode 'caffe' style, BGR
X = x_train.copy()
Y1 = preprocess_input(X)
del X # 메모리 삭제

#5: display image: x_train[0,:,:,:], Y1[0,:,:,:]
fig, ax = plt.subplots(1, 2, figsize=(10, 6))
##ax[0].imshow(x_train[0,:,:,:]/255.0)
ax[0].imshow(x_train[0,:,:,:].astype(np.uint8))
ax[0].set_title("x_train[0]:RGB")
ax[0].axis("off")

mean = np.array([103.939, 116.779, 123.68], dtype=np.float32) # ImageNet
Y1 -= mean
##ax[1].imshow(Y1[0,:,:,:]/255.0)
ax[1].imshow(Y1[0,:,:,:].astype(np.uint8))
ax[1].set_title("Y1[0]:BGR")
ax[1].axis("off")
fig.tight_layout()
plt.show()
```

사전학습 모델1: VGG 모델

VGG모델: 데이터 전처리

1. #1 에서 다양한 클래스로 구성된 이미지 데이터인 CIFAR-10 데이터셋을 로드한다. (CIFAR-10과 CIFAR-100이 있으며, -숫자는 데이터셋 구성하는 클래스 수 의미한다.)
한편, `tf.keras.applications.imagenet_utils` 에서 `preprocess_input()` 을 import 한다. `preprocess_input()` 함수는
입력 이미지의 채널 순서를 RGB 에서 BGR 순으로 변경하는 식으로 이미지를 전처리 한다.
2. #5는 같은 이미지에 대해서 채널 순서가 RGB 일 때 와 BGR 일 때 어떻게 다른 지 시각화 한다.

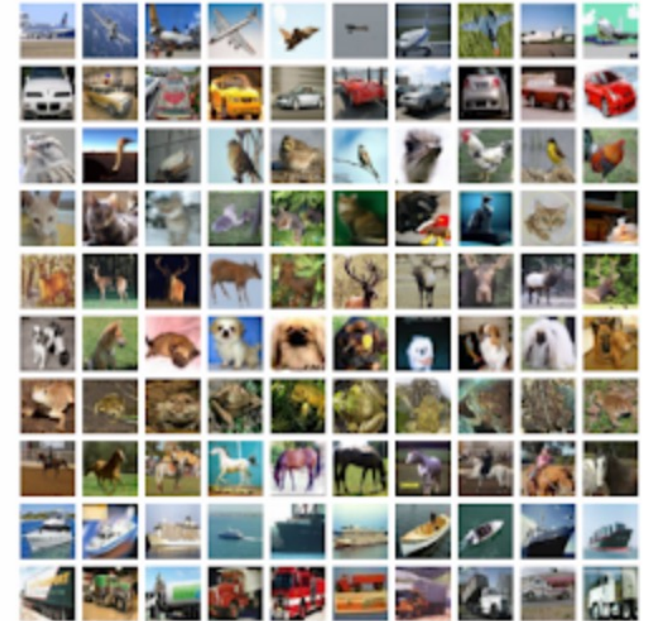
x_train[0]:RGB



Y1[0]:BGR



비행기
자동차
새
고양이
사슴
개
개구리
말
배
트럭



CIFAR-10 데이터 예시

사전학습 모델1: VGG 모델

VGG모델: 사전학습 가중치 이용한 분류(Classification)

```
ref1:
https://github.com/fchollet/deep-learning-models/releases/download/v0.1/vgg16_weights_tf_dim_ordering_tf_kernels.h5
ref2: # https://keras.io/applications/
'''

import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.applications import VGG16
from tensorflow.keras.applications.vgg16 import preprocess_input, decode_predictions
#from tensorflow.keras.applications.imagenet_utils import preprocess_input, decode_predictions
from tensorflow.keras.preprocessing import image # pip install pillow

#1:
##import tensorflow as tf
##gpus = tf.config.experimental.list_physical_devices('GPU')
##tf.config.experimental.set_memory_growth(gpus[0], True)

#2:
##W = 'C:/Users/user/.keras/models/vgg16_weights_tf_dim_ordering_tf_kernels.h5'
model = VGG16(weights='imagenet', include_top=True) # weights= W # VGG16 모델에 사전학습된 imagenet 가중치 불러오기
model.summary() # include_top=True 통해서 특징 추출 합성곱 층 아래에 완전연결 층 붙이기

#3: predict an image
img_path = './DATA/elephant.jpg' # './data/dog.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0) # (1, 224, 224, 3)
x = preprocess_input(x) # mode='caffe'
output = model.predict(x)

#3-1: (class_name, class_description, score)
print('Predicted:', decode_predictions(output, top=5)[0])
# decode_predictions 는 예측 결과 중에 최상위 5개(top=5) 항목 반환한다.

#4: display image and labels
plt.imshow(img)
plt.axis("off")
plt.show()
```


사전학습 모델1: VGG 모델

VGG모델: 사전학습 가중치 이용한 분류 결과

1/1 [=====] - 0s 305ms/step

Predicted: [('n02504013', 'Indian_elephant', 0.92946446), ('n02437312', 'Arabian_camel', 0.04909885), ('n01871265', 'tusker', 0.011825692), ('n02504458', 'African_elephant', 0.009556593), ('n02408429', 'water_buffalo', 1.7870392e-05)]



- ✓ decode_predictions() 를 사용하면 top= 에 주어진 값 갯수 만큼 모델의 최상위 예측 결과를 반환한다.
- 위 코끼리 이미지에 대해서 Indian_elephant, Arabian_camel, Tusker, African_elephant, 그리고 water_buffalo 순으로 모델이 분류한 것을 관찰할 수 있다.

사전학습 모델1: VGG 모델

VGG모델: 모델 구조만 사용해서, CIFAR-10 학습하기

```
1 import tensorflow as tf
2 from tensorflow.keras.datasets import cifar10 # cifar10 데이터셋 로드
3 from tensorflow.keras.applications.vgg16 import preprocess_input, VGG16
4 import numpy as np
5 import matplotlib.pyplot as plt
6
7 #1:
8 ##gpus = tf.config.experimental.list_physical_devices('GPU')
9 ##tf.config.experimental.set_memory_growth(gpus[0], True)
10
11 #2
12 (x_train, y_train), (x_test, y_test) = cifar10.load_data() # 데이터 로드
13 x_train = x_train.astype('float32') # (50000, 32, 32, 3) # 훈련 데이터 각 값을 실수 형태로 변환
14 x_test = x_test.astype('float32') # (10000, 32, 32, 3)
15
16 # one-hot encoding
17 y_train = tf.keras.utils.to_categorical(y_train) # 레이블 값 원핫 인코딩 벡터로 변환
18 y_test = tf.keras.utils.to_categorical(y_test)
19
20 #3: preprocessing, 'caffe', x_train, x_test: BGR
21 x_train = preprocess_input(x_train) # 훈련 데이터 전처리: 채널 순서를 RGB 에서 BGR 로 변경
22 x_test = preprocess_input(x_test)
23
24 #4: VGG16 에서, 사전학습된 가중치 사용하지 않음. 모델 전체 틀만 가져온다.
25 # include_top = True 사용해서 완전연결 층 불인 상태로 모델 로드한다.
26 # 분류할 클래스 갯수 = 10 개
27 # Input 의 shape = (32, 32, 3) 32*32 이미지, RGB 3차원
28 model = VGG16(weights=None, include_top=True, classes=10, input_shape=(32,32,3))
29 model.summary()
```

```
30
31 #5: train and evaluate the model
32 opt = tf.keras.optimizers.RMSprop(learning_rate=0.001)
33 model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
34 ret = model.fit(x_train, y_train, epochs=100, batch_size=400,
35               validation_split=0.2, verbose=0) #callbacks = [cp_callback]
36 train_loss, train_acc = model.evaluate(x_train, y_train, verbose=1)
37 test_loss, test_acc = model.evaluate(x_test, y_test, verbose=1)
38
39 #6: plot accuracy and loss
40 fig, ax = plt.subplots(1, 2, figsize=(10, 6))
41 ax[0].plot(ret.history['loss'], "g-")
42 ax[0].set_title("train loss")
43 ax[0].set_xlabel('epochs')
44 ax[0].set_ylabel('loss')
45
46 ax[1].plot(ret.history['accuracy'], "b-", label="train accuracy")
47 ax[1].plot(ret.history['val_accuracy'], "r-", label="val accuracy")
48 ax[1].set_title("accuracy")
49 ax[1].set_xlabel('epochs')
50 ax[1].set_ylabel('accuracy')
51 plt.legend()
52 fig.tight_layout()
53 plt.show()
```

- Imagenet 으로 사전학습된 가중치 사용하지 않고, 모델 구조 만 가져와서 from the scratch로 처음부터 학습시키자.

사전학습 모델1: VGG 모델

VGG모델: 전이학습(Transfer Learning) 통해 CIFAR-10 분류하기

```
1 import tensorflow as tf
2 from tensorflow.keras.datasets import cifar10
3 from tensorflow.keras.layers import Input, Dense, Flatten
4 from tensorflow.keras.applications.vgg16 import preprocess_input, VGG16
5 import numpy as np
6 import matplotlib.pyplot as plt
7
8 #1:
9 ##gpus = tf.config.experimental.List_physical_devices('GPU')
10 ##tf.config.experimental.set_memory_growth(gpus[0], True)
11
12 #2
13 (x_train, y_train), (x_test, y_test) = cifar10.load_data()
14 x_train = x_train.astype('float32') # (50000, 32, 32, 3)
15 x_test = x_test.astype('float32') # (10000, 32, 32, 3)
16
17 # one-hot encoding
18 y_train = tf.keras.utils.to_categorical(y_train) # 레이블 값 원핫 인코딩 벡터로 변환
19 y_test = tf.keras.utils.to_categorical(y_test)
20
21 # preprocessing, 'caffe', x_train, x_test: BGR
22 x_train = preprocess_input(x_train)
23 x_test = preprocess_input(x_test)
24
25 #3: resize_layer
26 inputs = Input(shape=(32, 32, 3))
27 resize_layer = tf.keras.layers.Lambda(lambda img: tf.image.resize(img, (224, 224)))(inputs)
28 # 32*32*3 shape 이미지를 input으로 받아서, 224*224*3 shape 으로 변환
29
30 #4: imagenet으로 사전학습된 가중치 불러오고, 완전연결 층은 붙이지 않는다. 곧, 합성곱 기반의 특징추출 층만 부른다.
31 vgg_model = VGG16(weights='imagenet', include_top=False,
32 input_tensor=resize_layer) # input_tensor= inputs, 224*224 사이즈 이미지를 input으로.
33 vgg_model.trainable=False # 특징 추출하는 backbone 부분 가중치 동결한다.
34
35 #4-1: output: classification
36 x = vgg_model.output # 특징 추출 층의 output. 곧, 합성곱 층 통과하면서 추출된 특징들을 x에 저장한다.
37 x = Flatten()(x) # 224*224 차원 벡터로 변환 후 x에 저장한다.
38 x = Dense(1024, activation='relu')(x) # x를 완전 연결 층에 넣는다.
39 outs = Dense(10, activation='softmax')(x) # 모델의 최종 출력이 outs 에 저장된다.
40 model = tf.keras.Model(inputs=inputs, outputs=outs)
41 model.summary()
42
```

```
43 #5: train and evaluate the model
44 filepath = "RES/ckpt/4404-model.h5"
45 cp_callback = tf.keras.callbacks.ModelCheckpoint(
46     filepath, verbose=0, save_best_only=True)
47 opt = tf.keras.optimizers.RMSprop(learning_rate=0.001)
48 model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
49 ret = model.fit(x_train, y_train, epochs=30, batch_size=64,
50     validation_split=0.2, verbose=0, callbacks=[cp_callback])
51 train_loss, train_acc = model.evaluate(x_train, y_train, verbose=2)
52 test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
53
54 #6: plot accuracy and loss
55 fig, ax = plt.subplots(1, 2, figsize=(10, 6))
56 ax[0].plot(ret.history['loss'], "g-")
57 ax[0].set_title("train loss")
58 ax[0].set_xlabel('epochs')
59 ax[0].set_ylabel('loss')
60
61 ax[1].set_ylim(0, 1.1)
62 ax[1].plot(ret.history['accuracy'], "b-", label="train accuracy")
63 ax[1].plot(ret.history['val_accuracy'], "r-", label="val accuracy")
64 ax[1].set_title("accuracy")
65 ax[1].set_xlabel('epochs')
66 ax[1].set_ylabel('accuracy')
67 plt.legend(loc='lower right')
68 fig.tight_layout()
69 plt.show()
70
```

사전학습 모델1: VGG 모델

VGG모델: 전이학습(Transfer Learning) 통해 CIFAR-100 분류하기

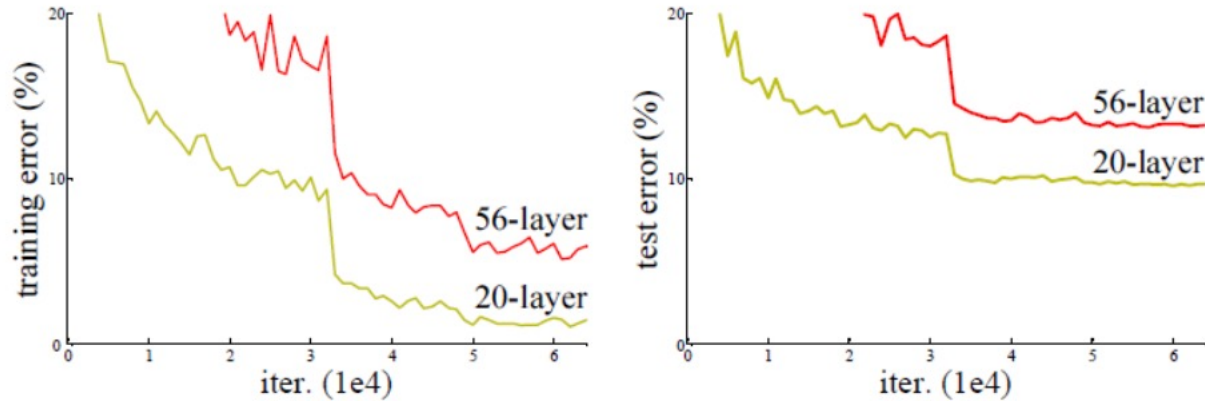
```
1 import tensorflow as tf
2 from tensorflow.keras.datasets import cifar100
3 from tensorflow.keras.layers import Input, Dense, Flatten, GlobalAveragePooling2D
4 from tensorflow.keras.applications.vgg16 import preprocess_input, VGG16
5 import numpy as np
6 import matplotlib.pyplot as plt
7
8 #1:
9 ##gpus = tf.config.experimental.List_physical_devices('GPU')
10 ##tf.config.experimental.set_memory_growth(gpus[0], True)
11
12 #2
13 (x_train, y_train), (x_test, y_test) = cifar100.load_data() # 'fine'
14
15 # one-hot encoding
16 y_train = tf.keras.utils.to_categorical(y_train)
17 y_test = tf.keras.utils.to_categorical(y_test)
18
19 # preprocessing, 'caffe', x_train, x_test: BGR
20 x_train = preprocess_input(x_train)
21 x_test = preprocess_input(x_test)
22
23 #3: resize_layer
24 inputs = Input(shape=(32, 32, 3))
25 resize_layer = tf.keras.layers.Lambda(lambda img: tf.image.resize(img, (224, 224)))(inputs)
26
27 #4:
28 vgg_model = VGG16(weights='imagenet', include_top=False,
29                    input_tensor=resize_layer) # input_tensor= inputs
30 vgg_model.trainable=False
31
32 #4-1: output: classification
33 x = vgg_model.output
34 x = Flatten()(x) # x = GlobalAveragePooling2D()(x)
35 x = Dense(1024, activation='relu')(x)
36 outs = Dense(100, activation='softmax')(x)
37 model = tf.keras.Model(inputs=inputs, outputs=outs)
38 ##model.summary()
```

```
40 #5: train and evaluate the model
41 filepath = "RES/ckpt/4505-model.h5"
42 cp_callback = tf.keras.callbacks.ModelCheckpoint(
43     filepath, verbose=0, save_best_only=True)
44 opt = tf.keras.optimizers.RMSprop(learning_rate=0.001)
45 model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
46 ret = model.fit(x_train, y_train, epochs=30, batch_size=32,
47               validation_split=0.2, verbose=0, callbacks = [cp_callback])
48 ##y_pred = model.predict(x_train)
49 ##y_label = np.argmax(y_pred, axis = 1)
50 ##C = tf.math.confusion_matrix(np.argmax(y_train, axis = 1), y_label)
51 train_loss, train_acc = model.evaluate(x_train, y_train, verbose=2)
52 test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
53
54 #6: plot accuracy and loss
55 fig, ax = plt.subplots(1, 2, figsize=(10, 6))
56 ax[0].plot(ret.history['loss'], "g-")
57 ax[0].set_title("train loss")
58 ax[0].set_xlabel('epochs')
59 ax[0].set_ylabel('loss')
60
61 ax[1].set_ylim(0, 1.1)
62 ax[1].plot(ret.history['accuracy'], "b-", label="train accuracy")
63 ax[1].plot(ret.history['val_accuracy'], "r-", label="val accuracy")
64 ax[1].set_title("accuracy")
65 ax[1].set_xlabel('epochs')
66 ax[1].set_ylabel('accuracy')
67 plt.legend(loc='lower right')
68 fig.tight_layout()
69 plt.show()
```

- trainable = True 로 설정하면 backbone의 가중치에도 학습이 반영되면서 합성곱 층 가중치를 ‘미세조정(Fine-tuning)’ 할 수 있다.
- 가중치를 미세 조정 할 때는 학습률을 보다 작게 설정한다. ch11 사전학습모델 실습자료

사전학습 모델2: ResNet 모델

- ✓ ResNet 모델: CNN 모델의 일종으로, 총 152개 층으로 구성되어 있다.
- ✓ ResNet 모델은 신경망이 깊어졌을 때 성능이 떨어지는 문제와, 가중치 소실을 해결한 모델이란 점에서 의의가 있다.
- 이론 상, 딥러닝 모델은 층이 깊어질 수록 training 데이터에 대해 특징들을 많이, 세밀하게 보게 되므로, 성능이 향상될 것이란 기대가 있다.
- 하지만, ResNet의 개발자들은 딥러닝 모델이 층을 무조건 많이 쌓는다고 해서 성능이 향상되지 않는다는 사실을 발견했다. 오히려, 상대적으로 얇은 신경망 모델보다, 깊은 신경망 모델에서 성능이 떨어지는 현상을 발견했다.



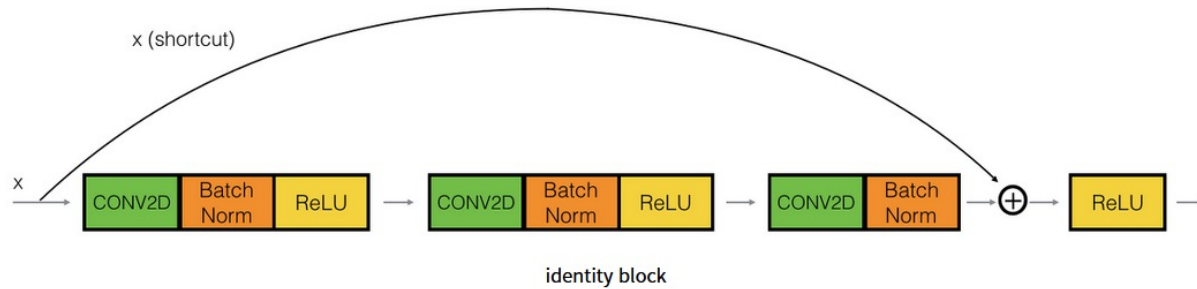
- 위 실험 결과를 보면 56개 층을 썼을 때, 20개 층을 썼을 때 보다 성능이 더 안 좋게 나온 것을 볼 수 있다. ResNet의 개발자들은 이런 현상을 방지하면서 동시에 가중치 소실 문제를 해결할 방법으로, ResNet 을 제안했다.

사전학습 모델2: ResNet 모델

ResNet 모델 구조

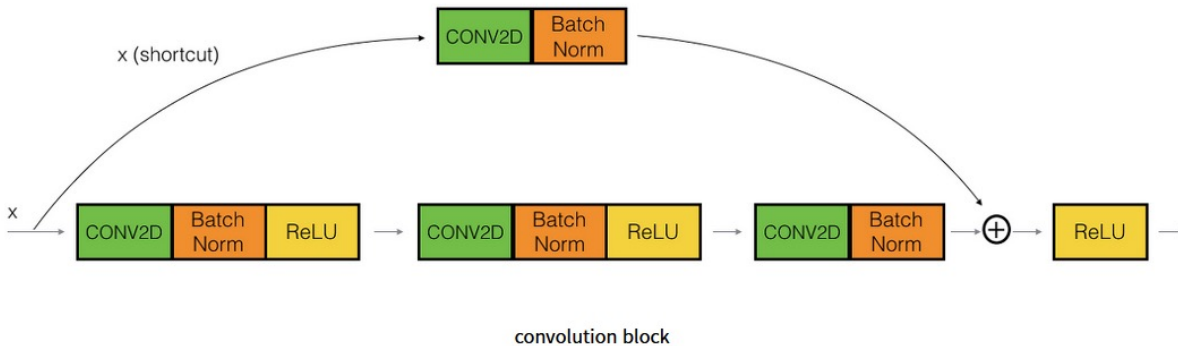
➤ ResNet 모델은 Identity block과 Convolution block 두 가지 Block 을 쌓은 형태다.

❖ Identity block



Identity block은 여러 가중치 층의 output에 입력 x 를 그대로 더해준다.

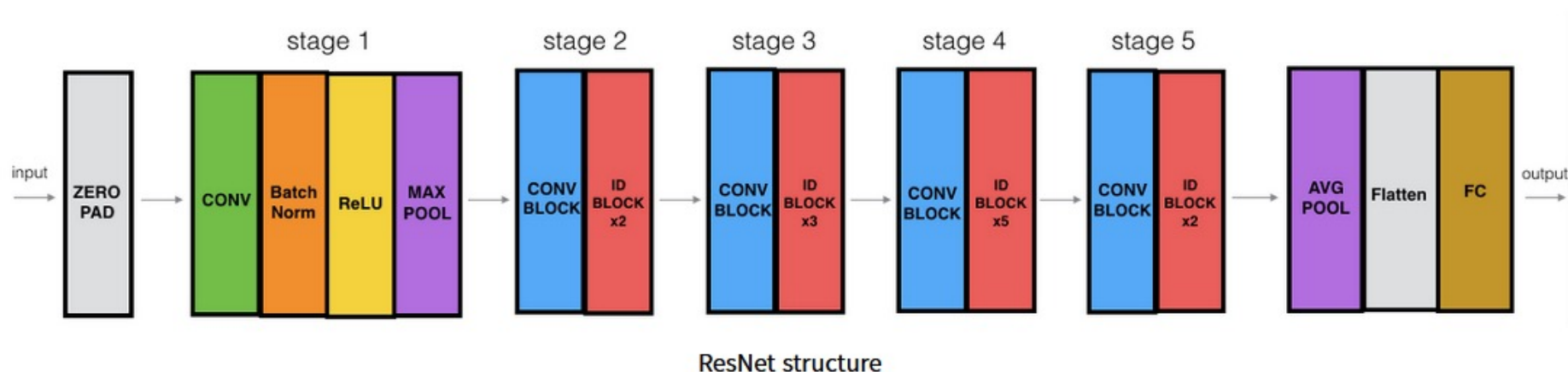
❖ Convolution block



Convolution block은 여러 가중치 층의 output과 입력 x에 1*1 합성곱 취한 것을 더한다.

사전학습 모델2: ResNet 모델

ResNet 모델 구조



- ResNet 모델은 위 두 가지 block을 위와 같은 방식으로 쌓아서 구성한다.
- Convolution block과 Identity block 두 가지를 교차해 가며 쌓은 구조임을 확인할 수 있다.

사전학습 모델2: ResNet 모델

ResNet: 사전학습 가중치를 이용한 분류

```
7 import numpy as np
8 import matplotlib.pyplot as plt
9 from tensorflow.keras.applications import ResNet50
10 from tensorflow.keras.applications.resnet import preprocess_input, decode_predictions
11 #from tensorflow.keras.applications.imagenet_utils import preprocess_input
12 from tensorflow.keras.preprocessing import image # pip install pillow
13
14 #1:
15 ##import tensorflow as tf
16 ##gpus = tf.config.experimental.list_physical_devices('GPU')
17 ##tf.config.experimental.set_memory_growth(gpus[0], True)
18
19 #2:
20 ##W = 'C:/Users/user/.keras/models/resnet50_weights_tf_dim_ordering_tf_kernels.h5'
21 model = ResNet50(weights='imagenet', include_top=True) # weights= W
22 model.summary()
23
24 #3: predict an image
25 #3-1
26 img_path = './DATA/dog.jpg' # './data/elephant.jpg' # './data/dog.jpg'
27 # img_path 에서 이미지 224*224 사이즈로 가져오기
28 img = image.load_img(img_path, target_size=(224, 224))
29 # 이미지를 넘파이 배열로 변환하기
30 x = image.img_to_array(img)
31 x = np.expand_dims(x, axis=0) # (1, 224, 224, 3), 배치차원 추가하기
32 x = preprocess_input(x) # 이미지 전처리 하기 (BGR) 순으로 채널 순서 변경
33 output = model.predict(x) # 모델 예측하기
34
35 #3-2: (class_name, class_description, score)
36 top5 = decode_predictions(output, top=5)[0]
37 print('Top-5 predicted:', top5)
38 #direct Top-1, and Top-5, ref[4502]
39
40 #4: display image and labels
41 plt.imshow(img)
42 plt.title(top5[0][1])
43 plt.axis("off")
44 plt.show()
45
```

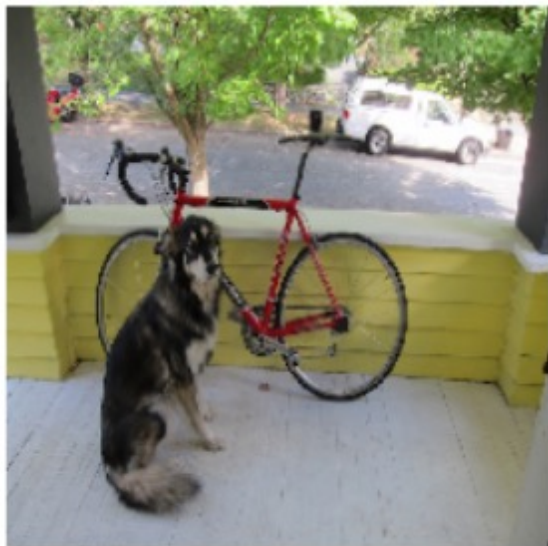

사전학습 모델2: ResNet 모델

ResNet50: 사전학습 가중치를 이용한 분류 결과

1/1 [=====] - 1s 988ms/step

Top-5 predicted: [('n02110063', 'malamute', 0.4726331), ('n02109961', 'Eskimo_dog', 0.2027256), ('n02110185', 'Siberian_husky', 0.08145712), ('n02088094', 'Afghan_hound', 0.039353408), ('n03218198', 'dogsled', 0.034458976)]

malamute



- 주어진 이미지에 대해, 모델이 ‘malamute’에 대해 가장 높은 확률을 부여한 것을 확인할 수 있다.

사전학습 모델2: ResNet 모델

ResNet50V2: 사전학습 가중치를 이용한 분류

```
7 import tensorflow as tf
8 from tensorflow.keras.applications import ResNet50V2
9 from tensorflow.keras.applications.resnet_v2 import preprocess_input, decode_predictions
10 from tensorflow.keras.preprocessing import image # pip install pillow
11 import numpy as np
12 import matplotlib.pyplot as plt
13
14 #1:
15 ##import tensorflow as tf
16 ##gpus = tf.config.experimental.list_physical_devices('GPU')
17 ##tf.config.experimental.set_memory_growth(gpus[0], True)
18
19 #2:
20 ##W = 'C:/Users/user/.keras/models/resnet50v2_weights_tf_dim_ordering_tf_kernels.h5'
21 model = ResNet50V2(weights='imagenet', include_top=True) # weights= W
22 model.summary()
23
24 #3: predict an image
25 #3-1:
26 img_path = './DATA/elephant.jpg' # './data/dog.jpg'
27 img = image.load_img(img_path, target_size=(224, 224))
28 x = image.img_to_array(img)
29 x = np.expand_dims(x, axis=0) # (1, 224, 224, 3)
30 x = preprocess_input(x)
31 ##x = tf.keras.applications.imagenet_utils.preprocess_input(x, mode='tf')
32 output = model.predict(x)
33
34 #3-2: (class_name, class_description, score)
35 top5 = decode_predictions(output, top=5)[0]
36 print('Top-5 predicted:', top5)
37 #direct Top-1, and Top-5, ref[4502]
38
39 #4: display image and labels
40 plt.imshow(img)
41 plt.title(top5[0][1])
42 plt.axis("off")
43 plt.show()
```

사전학습 모델2: ResNet 모델

ResNet50V2: 사전학습 가중치를 이용한 분류

1/1 [=====] - 1s 1s/step

Top-5 predicted: [('n02504013', 'Indian_elephant', 0.9769355), ('n02504458', 'African_elephant', 0.011323931), ('n01871265', 'tusker', 0.0076959813), ('n02437312', 'Arabian_camel', 0.0036789305), ('n01704323', 'triceratops', 0.0003181935)]

Indian_elephant



➤ 주어진 이미지에 대해, 모델이 ‘Indian elephant’에 대해 가장 높은 확률을 부여한 것을 확인할 수 있다.

사전학습 모델2: ResNet 모델

ResNet50 전이학습: CIFAR-10 분류

```
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.layers import Input, Dense, GlobalAveragePooling2D
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.applications.resnet import preprocess_input, decode_predictions
from tensorflow.keras.preprocessing import image # pip install pillow

import numpy as np
import matplotlib.pyplot as plt

#1:
##import tensorflow as tf
##gpus = tf.config.experimental.list_physical_devices('GPU')
##tf.config.experimental.set_memory_growth(gpus[0], True)

#2
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
x_train = x_train.astype('float32') # (50000, 32, 32, 3)
x_test = x_test.astype('float32') # (10000, 32, 32, 3)

# one-hot encoding
y_train = tf.keras.utils.to_categorical(y_train)
y_test = tf.keras.utils.to_categorical(y_test)

# preprocessing, 'caffe', x_train, x_test: BGR
x_train = preprocess_input(x_train)
x_test = preprocess_input(x_test)

#3: resize_layer
inputs = Input(shape = (32, 32, 3))
resize_layer = tf.keras.layers.Lambda(lambda img: tf.image.resize(img, (224, 224)))(inputs)
res_model = ResNet50(weights = 'imagenet', include_top = False,
                      input_tensor = inputs)#resize_layer # inputs # 특징 추출 층만 사용
res_model.trainable=False # backbone 가중치 동결

#4: create top for cifar10 classification
x = res_model.output
x = GlobalAveragePooling2D()(x) # 2차원 데이터에 대해 평균 풀링
x = Dense(1024, activation = 'relu')(x)
outs = Dense(10, activation = 'softmax')(x)
model = tf.keras.Model(inputs = inputs, outputs=outs)
model.summary()
```

```
#5: train and evaluate the model
filepath = "RES/ckpt/4603-model.h5"
cp_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath, verbose = 0, save_best_only = True)

opt = tf.keras.optimizers.RMSprop(learning_rate = 0.001)
model.compile(optimizer = opt, loss = 'categorical_crossentropy', metrics = ['accuracy'])
ret = model.fit(x_train, y_train, epochs = 30, batch_size = 32,
                validation_split = 0.2, verbose = 1, callbacks = [cp_callback])

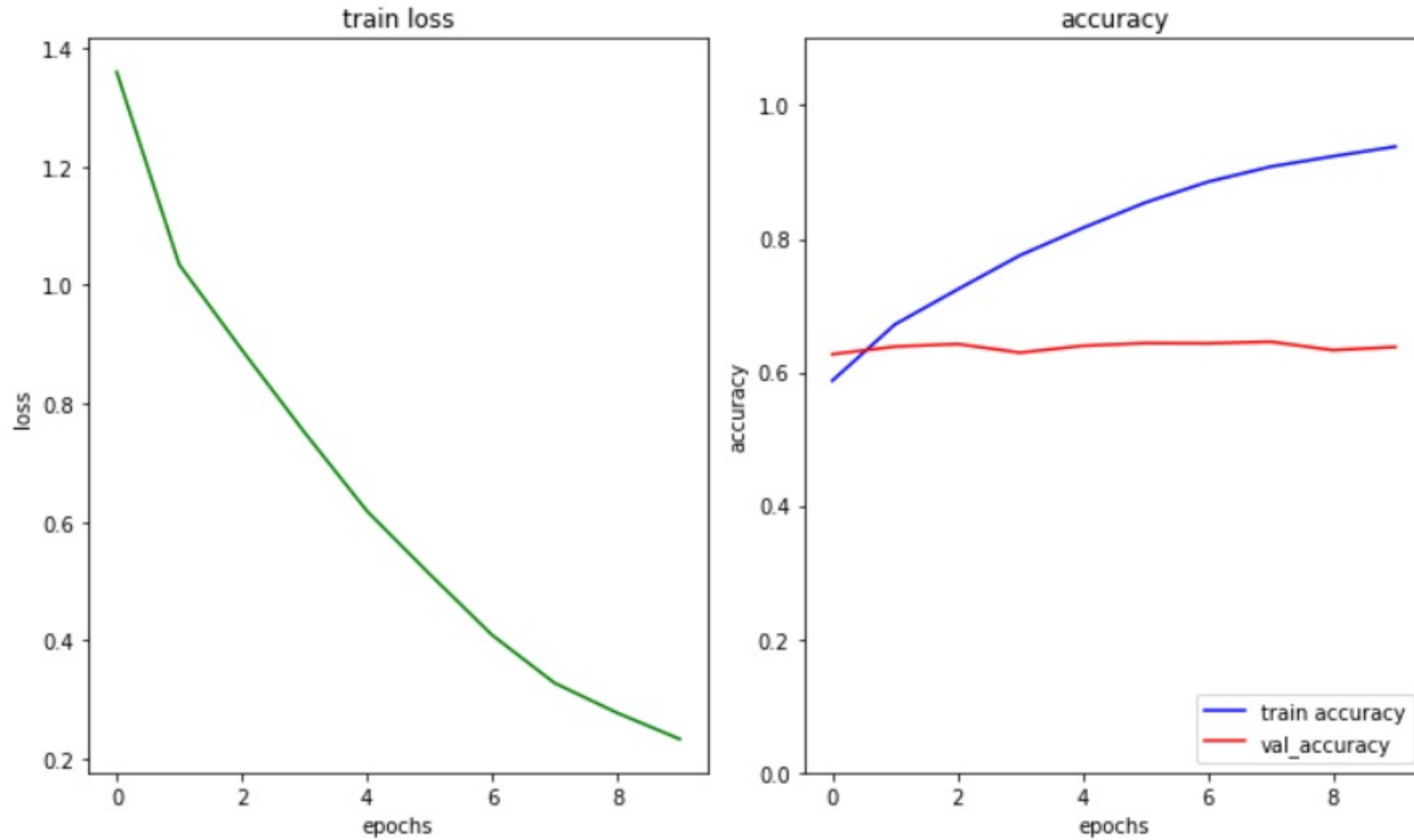
#6: plot accuracy and loss
fig, ax = plt.subplots(1, 2, figsize = (10, 6))
ax[0].plot(ret.history['loss'], "g-")
ax[0].set_title("train loss")
ax[0].set_xlabel('epochs')
ax[0].set_ylabel('loss')

ax[1].set_ylim(0, 1.1)
ax[1].plot(ret.history['accuracy'], "b-", label = "train accuracy")
ax[1].plot(ret.history['val_accuracy'], "r-", label = "val_accuracy")
ax[1].set_title("accuracy")
ax[1].set_xlabel('epochs')
ax[1].set_ylabel('accuracy')
plt.legend(loc = 'lower right')
fig.tight_layout()
plt.show()
```

- Resize layer 는 모델 성능을 높이기 위해 이미지를 32*32 에서 224*224 사이즈로 확대한다.
- ResNet50 모델에 대해, imagenet 에 사전학습 된 가중치를 적용한다. 분류를 위한 완전연결층은 함께 가져오지 않으며,
- 모델 학습 시킬 땐 모델 가장 뒷단에 새로 붙인, 완전연결 층 2개의 가중치만 학습시킨다.

사전학습 모델2: ResNet 모델

ResNet50 전이학습: CIFAR-10 분류 결과



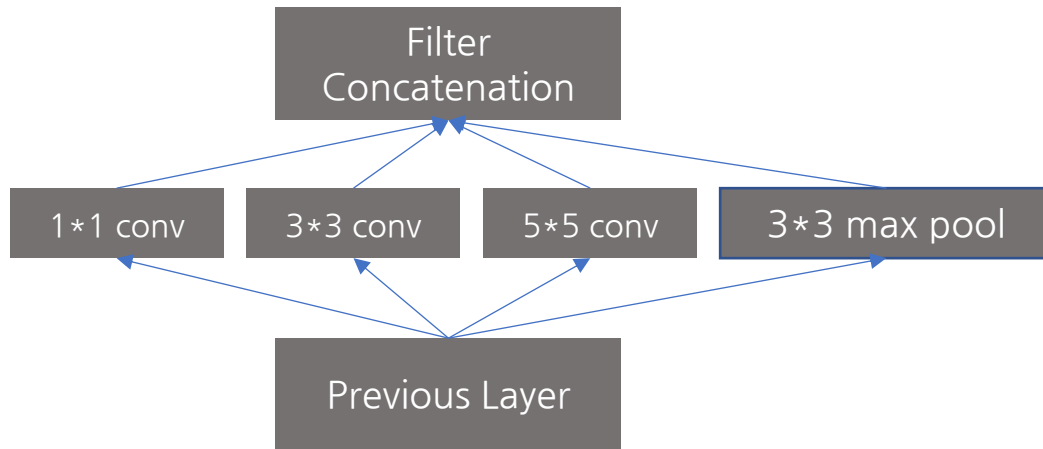
- 훈련 결과, 훈련 데이터 정확도는 97.76% 이다. 한편, 테스트 데이터 정확도는 90.95% 이다.
- 만약 ResourceExhaustError 오류가 발생하면, 메모리가 부족한 것이므로 batch_size=16 또는 8로 배치 크기를 감소시켜 실행해 보자.

사전학습 모델3: Inception과 GoogLeNet 모델

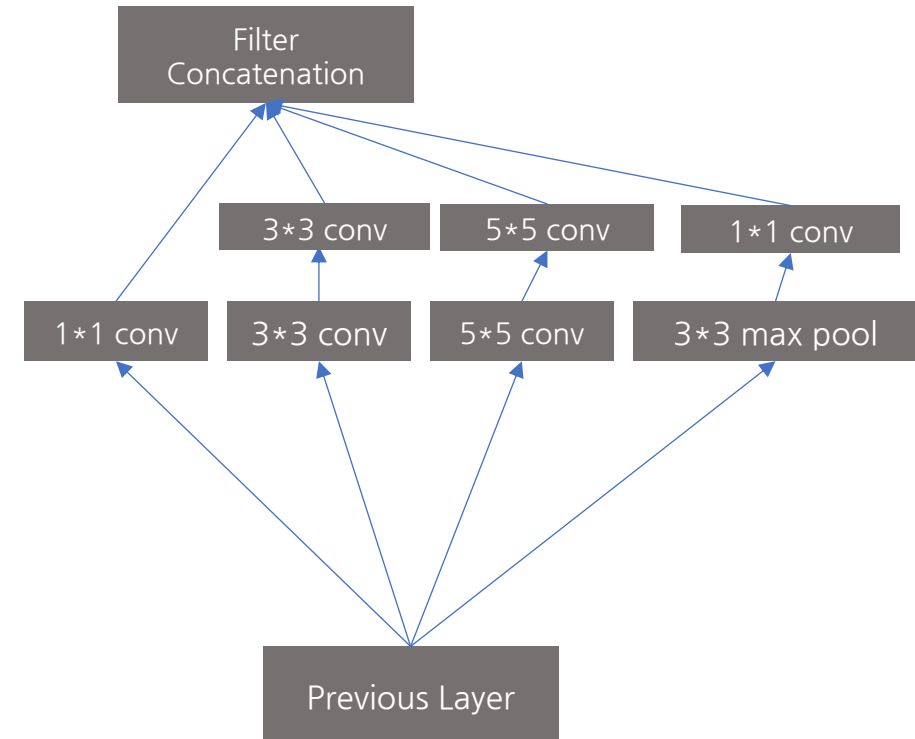
Inception & GoogLeNet

- GoogLeNet : 인셉션(Inception) 모듈을 쌓아서 구성하는 방식의, 합성곱 신경망(CNN) 모델이다.
- 인셉션 모듈은 한 층에서 하나의 합성곱 필터를 쌓는 것이 아니라, 여러 개의 서로 다른 합성곱 층 또는 풀링을 수행한 후에 병합한다.

구글이 사용했던 인셉션 모듈 1개 예시



(a) Inception module, naïve version



(b) Inception module with dimension reduction

사전학습 모델3: Inception과 GoogLeNet 모델

1-인셉션 모듈: CIFAR-10 분류

```
5 import tensorflow as tf
6 from tensorflow.keras.datasets import cifar10
7 from tensorflow.keras.layers import Input, Conv2D, Dense
8 from tensorflow.keras.layers import Flatten, MaxPooling2D, GlobalAveragePooling2D
9 import numpy as np
10 import matplotlib.pyplot as plt
11
12 #1:
13 ##import tensorflow as tf
14 ##gpus = tf.config.experimental.list_physical_devices('GPU')
15 ##tf.config.experimental.set_memory_growth(gpus[0], True)
16
17 #2: CIFAR-10 이미지 데이터 로드
18 (x_train, y_train), (x_test, y_test) = cifar10.load_data()
19 # 이미지 데이터 텐서 하나하나 구성하는 스칼라들을 모두 실수로 변환
20 x_train = x_train.astype('float32') # (50000, 32, 32, 3)
21 x_test = x_test.astype('float32') # (10000, 32, 32, 3)
22 # 실수 스칼라 값들 모두 0과 1 사이 값으로 정규화
23 x_train = x_train / 255.0
24 x_test = x_test / 255.0
25
26 # one-hot encoding
27 # train 데이터의 레이블과 test 데이터 레이블을 모두 원핫 인코딩 벡터로 변환
28 y_train = tf.keras.utils.to_categorical(y_train)
29 y_test = tf.keras.utils.to_categorical(y_test)
30
31 #3: simple Inception_layer
32 # input 사이즈 지정(32, 32, 3) : 32*32 사이즈, RGB 3채널
33 inputs = Input(shape=(32, 32, 3))
34 # 64 개 필터 사용, 윈도우 사이즈 1*1, padding=same 지정해서 층 입력에 대해 zero padding 적용해서 입력과 출력사이즈
35 # 같아지게 만든다. 활성화함수는 relu 사용, 층 이름은 L1으로 지정
36 L1 = Conv2D(64, (1,1), padding='same', activation='relu', name="L1")(inputs)
37 L2 = Conv2D(64, (3,3), padding='same', activation='relu', name="L2")(L1)
38
39 L3 = Conv2D(64, (1,1), padding='same', activation='relu')(inputs)
40 L3 = Conv2D(64, (5,5), padding='same', activation='relu', name="L3")(L3)
41
42 # 3*3 윈도우로 2차원 데이터에 대해 MaxPooling 적용, strides=(height stride, width stride)
43 L4 = MaxPooling2D((3,3), strides=(1,1), padding='same')(inputs)
44 L4 = Conv2D(64, (1,1), padding='same', activation='relu', name="L4")(L4)
45 # axis=3 이므로, 3차원으로 모든 같은 shape인 L1, L2, L3, L4 를 '붙인다'.
46 output = tf.keras.layers.concatenate([L1, L2, L3, L4], axis = 3)
47
48 #4: create top for cifar10 classification
49 output = Flatten()(output) # 벡터로 변환한다.
50 outs = Dense(10, activation='softmax')(output)
51 model = tf.keras.Model(inputs=inputs, outputs=outs)
52 model.summary()
```

```
54 #5: train and evaluate the model
55 opt = tf.keras.optimizers.RMSprop(learning_rate=0.001)
56 model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
57 ret = model.fit(x_train, y_train, epochs=30, batch_size=128,
58               validation_split=0.2, verbose=0)
59
60 #6: plot accuracy and loss
61 fig, ax = plt.subplots(1, 2, figsize=(10, 6))
62 ax[0].plot(ret.history['loss'], "g-")
63 ax[0].set_title("train loss")
64 ax[0].set_xlabel('epochs')
65 ax[0].set_ylabel('loss')
66
67 ax[1].set_ylim(0, 1.1)
68 ax[1].plot(ret.history['accuracy'], "b-", label="train accuracy")
69 ax[1].plot(ret.history['val_accuracy'], "r-", label="val accuracy")
70 ax[1].set_title("accuracy")
71 ax[1].set_xlabel('epochs')
72 ax[1].set_ylabel('accuracy')
73 plt.legend(loc='lower right')
74 fig.tight_layout()
75 plt.show()
```

- 위 코드는 'Inception module with dimension reduction' 구조, 다시 말해 1개 Inception 모듈을 구현하고, CIFAR-10 데이터셋으로 훈련 후 테스트 한 것이다.
- 인셉션 모듈 위에 분류를 위해 완전연결 층 1개를 붙였다.

사전학습 모델3: Inception과 GoogLeNet 모델

InceptionV3, InceptionResNetV2, Xception: 사전학습 가중치를 이용한 분류

```
12 from tensorflow.keras.applications import InceptionV3, InceptionResNetV2, Xception
13 from tensorflow.keras.applications.imagenet_utils import preprocess_input
14 from tensorflow.keras.applications.imagenet_utils import decode_predictions
15 from tensorflow.keras.preprocessing import image # pip install pillow
16 import numpy as np
17 import matplotlib.pyplot as plt
18
19 #1:
20 ##import tensorflow as tf
21 ##gpus = tf.config.experimental.list_physical_devices('GPU')
22 ##tf.config.experimental.set_memory_growth(gpus[0], True)
23
24
25 #2: imagenet으로 훈련시킨 사전학습 가중치 적용한, inceptionv3, inceptionresnetv2, xception
26 # 모델 불러온다. 모두 완전연결층도 위에 붙인 상태로, 전체 모델을 불러온다.
27 model1 = InceptionV3(weights='imagenet', include_top=True)
28 model2 = InceptionResNetV2(weights='imagenet', include_top=True)
29 model3 = Xception(weights='imagenet', include_top=True)
30 #model1.summary()
31 #model2.summary()
32 #model3.summary()
33
34 #3: predict an image
35 #3-1:
36 img_path = './DATA/elephant.jpg' # './data/dog.jpg'
37 img = image.load_img(img_path, target_size=(299, 299)) # 이미지를 299*299 사이즈로 불러온다.
38 x = image.img_to_array(img) # 이미지를 넘파이 배열로 변환한다.
39 x = np.expand_dims(x, axis=0) # (1, 299, 299, 3) # 이미지 shape 에 배치차원을 추가한다.
40 x = preprocess_input(x) # 이미지를 전처리 한다.
41 output1 = model1.predict(x) # 이미 학습된 상태의 InceptionV3 모델로 input 이미지를 예측한다.
42 output2 = model2.predict(x) # 이미 학습된 상태의 InceptionResNetV2 모델로 input 이미지를 예측한다.
43 output3 = model3.predict(x) # 이미 학습된 상태의 xception 모델로 input 이미지를 예측한다.
44
45 #3-2: (class_name, class_description, score)
46 top5 = decode_predictions(output1, top=5)[0] # 모델 예측 결과 중 최상위 5개만 추려내서 top5에 저장한다.
47 print('InceptionV3, Top-5:', top5) # 인셉션v3 모델의 최상위 5개 결과 출력
48
49 top5 = decode_predictions(output2, top=5)[0]
50 print('InceptionResNetV2, Top-5:', top5) #인셉션 레스넷v2 모델의 최상위 5개 결과 출력
51
52 top5 = decode_predictions(output3, top=5)[0]
53 print('Xception, Top-5:', top5) # xception 모델의 최상위 5개 결과 출력
54
55 #4: display image and labels
56 plt.imshow(img)
57 plt.title(top5[0][1])
58 plt.axis("off")
59 plt.show()
```

- 왼쪽 코드는 ‘imagenet’ 에서 미리 학습된 가중치를 사용하고, Include_top = True 로 분류를 위한 완전 연결층을 포함하는 InceptionV3 모델, InceptionResNetV2 모델, Xception 모델 을 생성한다.
- 입력층은 299*299*3 shape의 이미지를 내놓는다. 최종 출력 층(=마지막 완전연결층) 은 imagenet 분류를 위해 1,000 개 히든유닛을 갖는다.

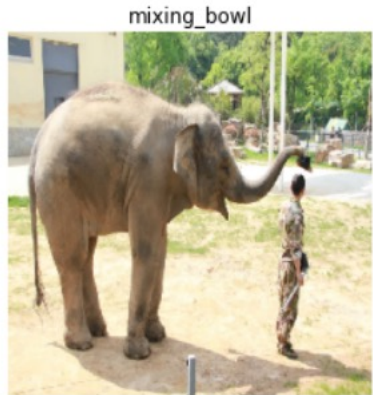
사전학습 모델3: Inception과 GoogLeNet 모델

InceptionV3, InceptionResNetV2, Xception: 사전학습 가중치를 이용한 분류

InceptionV3, Top-5: [(`'n03950228'`, `'pitcher'`, 0.72197545), (`'n01924916'`, `'flatworm'`, 0.27800676), (`'n04522168'`, `'vase'`, 1.5433465e-05), (`'n03908714'`, `'pencil_sharpener'`, 1.8314827e-06), (`'n04131690'`, `'saltshaker'`, 3.063997e-07)]

InceptionResNetV2, Top-5: [(`'n06596364'`, `'comic_book'`, 1.0), (`'n02791124'`, `'barber_chair'`, 7.7977665e-29), (`'n04517823'`, `'vacuum'`, 2.2254776e-32), (`'n15075141'`, `'toilet_tissue'`, 0.0), (`'n02342885'`, `'hamster'`, 0.0)]

Xception, Top-5: [(`'n03775546'`, `'mixing_bowl'`, 0.6560512), (`'n03825788'`, `'nipple'`, 0.16745721), (`'n03763968'`, `'military_uniform'`, 0.11219646), (`'n03942813'`, `'ping-pong_ball'`, 0.050320935), (`'n02704792'`, `'amphibian'`, 0.0033987707)]



- InceptionV3 모델의 최상위 예측 결과는 ‘pitcher’(72.19%)였다.
- InceptionResNetV2 모델의 최상위 예측 결과는 ‘comic book’(100%) 이다.
- Xception 모델의 최상위 예측 결과는 ‘mixing bowl’(65.6%) 이다.